Code Documentation

Project Name: AI Patrol and Chase System in Unity

- **FSM (Finite State Machine)**
- **Sensory system (detection via raycast & distance)**
- **Patrolling, chasing, searching**
- **Comments** for every section so even beginners can understand

```
using UnityEngine;
using UnityEngine.AI;

public class AIStateMachine : MonoBehaviour
{
    // Reference to the player
    public Transform player;

    // NavMeshAgent helps the AI move using Unity's pathfinding
    public NavMeshAgent agent;

    // AI Vision/Detection Settings
    public float detectionRange = 10f;     // How far AI can see
    public float fieldOfView = 120f;       // Field of view angle
    public float searchDuration = 3f;      // How long AI searches after
losing sight

    // Waypoints for patrol (set in inspector)
    public Transform[] patrolPoints;
    private int currentPatrolIndex = 0;

    // FSM States
    private enum AIState { Patrolling, Chasing, Searching }
    private AIState currentState;

    // Memory of where the player was last seen
    private Vector3 lastSeenPosition;
```

```
    // Timers
    private float searchTimer = 0f;

    void Start()
    {
        currentState = AIState.Patrolling;
        agent = GetComponent<NavMeshAgent>();
        GoToNextPatrolPoint();
    }

    void Update()
    {
        // Update FSM each frame
        switch (currentState)
        {
            case AIState.Patrolling:
                Patrol();
                break;

            case AIState.Chasing:
                Chase();
                break;

            case AIState.Searching:
                Search();
                break;
        }

        // Check if player is in sight every frame
        if (CanSeePlayer())
        {
            currentState = AIState.Chasing;
            lastSeenPosition = player.position;
        }
    }

    // ----------------- FSM Behaviors --------------------

    // Patrol between waypoints
    void Patrol()
    {
        if (patrolPoints.Length == 0) return;
```

```
        // Go to next point if current is reached
        if (!agent.pathPending && agent.remainingDistance < 0.5f)
        {
            GoToNextPatrolPoint();
        }
    }

    // Chase the player directly
    void Chase()
    {
        agent.SetDestination(player.position);

        float distance = Vector3.Distance(transform.position,
player.position);
        if (distance > detectionRange || !CanSeePlayer())
        {
            currentState = AIState.Searching;
            searchTimer = 0f;
        }
    }

    // Search the last known player location
    void Search()
    {
        agent.SetDestination(lastSeenPosition);

        searchTimer += Time.deltaTime;

        // After time runs out, return to patrol
        if (searchTimer > searchDuration)
        {
            currentState = AIState.Patrolling;
            GoToNextPatrolPoint();
        }
    }

    // ----------------- Utility Functions --------------------

    // Move to the next waypoint
    void GoToNextPatrolPoint()
    {
        if (patrolPoints.Length == 0) return;
```

```
        agent.destination = patrolPoints[currentPatrolIndex].position;
        currentPatrolIndex = (currentPatrolIndex + 1) %
patrolPoints.Length;
    }

    // Check if the AI can see the player using raycasting and FOV
    bool CanSeePlayer()
    {
        Vector3 directionToPlayer = player.position - transform.position;
        float angle = Vector3.Angle(directionToPlayer, transform.forward);

        if (directionToPlayer.magnitude < detectionRange && angle <
fieldOfView / 2)
        {
            Ray ray = new Ray(transform.position + Vector3.up,
directionToPlayer.normalized);
            RaycastHit hit;

            if (Physics.Raycast(ray, out hit, detectionRange))
            {
                if (hit.transform == player)
                {
                    return true;
                }
            }
        }

        return false;
    }
}
```